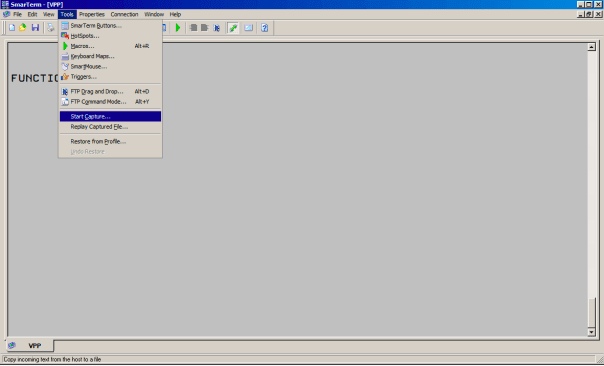
**Background**

As part of our lab accreditation requirements, we have to provide measurement uncertianty estimates for all tests at all hospital sites. As you might imagine, with thousands of testcodes in [Sunquest LIS](https://www.sunquestinfo.com/), getting all the coefficients of variation (CVs) represents a daunting task for the quality technologist to accomplish. As it turns out, by capturing the ssh session in a .txt file, you can use R’s [dplyr](https://dplyr.tidyverse.org/) package to do this all in few lines of code.

**Getting the Raw Data**

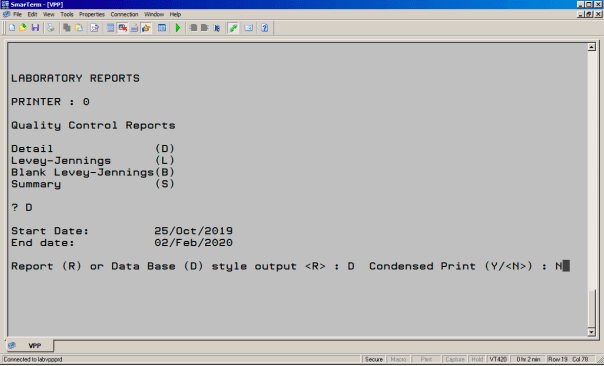
You need to get the raw data from Sunquest. You can capture the telnet (yes… older versions of Sunquest use telnet and pass patient information and user passwords unencrypted across the hospital network o\_O) or the ssh session to a file using the [Esker SmarTerm](https://www.esker.com/business-process-solutions/automation-products/host-access-terminal-emulation-software/) which Sunquest packages in their product and refers to as “roll-n-scroll”. People disparriage SmarTerm as an old “dos tool”–whereas Sunquest is hosted on [AIX operating system](https://en.wikipedia.org/wiki/IBM_AIX). SmarTerm access to Sunquest is a gagillion times faster than the GUI and permits us to capture the raw QC data we need. To capture the session select from the dropdown menu as shown here:



If you are using Mac OS or Linux OS, you can also capture the ssh session by connecting from the terminal and using tee to dump the session to a file.

ssh user@serverIPaddress | tee captured\_session.txt

Once you have connected, use the QC function and select output printer 0 (meaning the screen) and make these selections, changing the dates as appropriate:



If you make no selections at all for any of:

* TEST:
* WORKSHEET:
* METHOD:
* CONTROL:
* SHIFT #:
* TECH:
* TESTS REQUESTED:

then you will extract everything, which is what you want and which will make for a very big .txt file. There will be a delay and then thousands of QC results will dump to the screen and to your file. When this is complete, end your SmarTerm or ssh or telnet (cringe) session. I saved my text dump as raw\_SQ8.txt.

**Getting it intro R and parsing it**

Your data will come out as a fixed with file with no delimiters. It will also have a bunch of junk at the bottom and top of the file detailing your commands from the start and end of the session. These need to be discarded. I just used grep() to find all the lines with the appropriate date pattern. After reading it in, because I am lazy, I wrote it back out and read it in again with read.fwf()

library(tidyverse)

library(lubridate)

library(knitr)

# Note to my friend SK - yes... this is mostly in base-R...

# create a connection

con <- file(file.path("raw\_SQ8.txt"))

raw.qc.data <- readLines(con)

close(con)

#find good rows

good.data <- grep("[0-9]{2}(Jan|Feb|Mar|Apr|May|Jun|Jul|Aug|Sep|Oct|Nov|Dec)[2][0][0-9]{6}",raw.qc.data)

raw.qc.data <- raw.qc.data[good.data]

#remove a screwball encoding character

raw.qc.data[1] <- substr(raw.qc.data[1],6,nchar(raw.qc.data[1]))

con <- file("temp.txt")

#rewrite the file with no garbage in it.

writeLines(raw.qc.data, con)

close(con)

raw.qc.data <- read.fwf("temp.txt",c(6,6,6,20,13,6,2,15,100))

file.remove("temp.txt")

names(raw.qc.data) <- c("test.code","instr.code","[qc.name](http://qc.name)","qc.expire",

"date.performed","tech.code","shift",

"result","modifier")

raw.qc.data <- data.frame(lapply(raw.qc.data, trimws))

raw.qc.data$result <- as.numeric(as.character(raw.qc.data$result))

raw.qc.data$date.performed <- dmy\_hm(raw.qc.data$date.performed)

raw.qc.data$tech.code <- as.numeric(raw.qc.data$tech.code) #anonymize tech codes

raw.qc.data <- arrange(raw.qc.data, instr.code, test.code)

Now that all the data munging is done, we can examine the data:

| **test.code** | **instr.code** | [**qc.name**](http://qc.name) | **qc.expire** | **date.performed** | **tech.code** | **shift** | **result** | **modifier** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BCL | JBGAS | RAD1 | R0173 EXP MAR 2021 | 2019-11-15 09:17:00 | 68 | 2 | 122 | NA |
| BCL | JBGAS | RAD1 | R0173 EXP MAR 2021 | 2019-11-15 20:51:00 | 68 | 3 | 122 | NA |
| BCL | JBGAS | RAD1 | R0173 EXP MAR 2021 | 2019-11-15 21:47:00 | 68 | 3 | 122 | NA |
| BCL | JBGAS | RAD1 | R0173 EXP MAR 2021 | 2019-11-15 21:50:00 | 68 | 3 | 122 | NA |
| BCL | JBGAS | RAD1 | R0173 EXP MAR 2021 | 2019-11-17 07:10:00 | 15 | 1 | 122 | NA |
| BCL | JBGAS | RAD1 | R0173 EXP MAR 2021 | 2019-11-17 07:11:00 | 15 | 1 | 122 | NA |

And finally, we can make the dplyr magic happen and discard results for which the counts are too small, which I have chosen to be <20:

raw.qc.data %>% dplyr::filter(![is.na](http://is.na)(result)) %>%

group\_by(instr.code,test.code,[qc.name](http://qc.name),qc.expire) %>%

summarise(median = median(result),

IQR = IQR(result),

mean = mean(result),

SD = sd(result),

min = min(result),

max = max(result),

CV = round(sd(result, na.rm = TRUE)/mean(result, na.rm = TRUE)\*100,2),

count = n()) %>%

filter(count ≥ 20) %>%

arrange(instr.code, test.code, median) -> summary.table

Which gives us output like this:

head(summary.table)

| **instr.code** | **test.code** | [**qc.name**](http://qc.name) | **qc.expire** | **median** | **IQR** | **mean** | **SD** | **min** | **max** | **CV** | **count** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| JBGAS | BCL | RAD3 | R0141 EXP SEP 2017 | 65.0 | 1.000 | 65.145454 | 0.6503043 | 63.0 | 66.0 | 1.00 | 55 |
| JBGAS | BCL | RAD2 | R0175 EXP MAR 2021 | 97.0 | 0.000 | 97.128205 | 0.3364820 | 97.0 | 98.0 | 0.35 | 78 |
| JBGAS | BCL | RAD1 | R0173 EXP MAR 2021 | 122.0 | 0.000 | 122.122807 | 0.5691527 | 121.0 | 124.0 | 0.47 | 57 |
| JBGAS | BGLUC | RAD1 | R0173 EXP MAR 2021 | 1.5 | 0.000 | 1.507017 | 0.0257713 | 1.5 | 1.6 | 1.71 | 57 |
| JBGAS | BGLUC | RAD2 | R0175 EXP MAR 2021 | 5.6 | 0.075 | 5.585897 | 0.0639081 | 5.4 | 5.7 | 1.14 | 78 |
| JBGAS | BGLUC | RAD3 | R0141 EXP SEP 2017 | 13.7 | 0.100 | 13.763636 | 0.1310409 | 13.4 | 14.1 | 0.95 | 55 |

This permits us to toss out results with low counts. But what about handling outliers? Well, we can calculate the z-scores of the raw data by joining the the mean and SD results back to the raw data.

raw.qc.data %>%

left\_join(select(summary.table,c(instr.code:qc.expire, mean, SD)),

by = c("test.code","instr.code", "[qc.name](http://qc.name)", "qc.expire")) %>%

mutate(z.score = (result - mean)/SD) -> raw.qc.data

This will permit you to suppress results outside a certain z-score. So, let’s suppress all results with an undefined z-score and all results with a z-score >= 4:

raw.qc.data %>%

drop\_na(z.score) %>%

filter(abs(z.score) < 4) -> raw.qc.data

Now , we can re-run the dplyr summary:

raw.qc.data %>% dplyr::filter(![is.na](http://is.na)(result)) %>%

group\_by(instr.code,test.code,[qc.name](http://qc.name),qc.expire) %>%

summarise(median = median(result),

IQR = IQR(result),

mean = mean(result),

SD = sd(result),

min = min(result),

max = max(result),

CV = round(sd(result, na.rm = TRUE)/mean(result, na.rm = TRUE)\*100,2),

count = n()) %>%

filter(count ≥ 20) %>%

arrange(instr.code, test.code, median) -> summary.table.no.outliers

And now we have a summary of every QC CV in our Sunquest system with outliers suppressed:

head(summary.table.no.outliers)

| **instr.code** | **test.code** | [**qc.name**](http://qc.name) | **qc.expire** | **median** | **IQR** | **mean** | **SD** | **min** | **max** | **CV** | **count** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| JBGAS | BCL | RAD3 | R0141 EXP SEP 2017 | 65.0 | 1.000 | 65.145454 | 0.6503043 | 63.0 | 66.0 | 1.00 | 55 |
| JBGAS | BCL | RAD2 | R0175 EXP MAR 2021 | 97.0 | 0.000 | 97.128205 | 0.3364820 | 97.0 | 98.0 | 0.35 | 78 |
| JBGAS | BCL | RAD1 | R0173 EXP MAR 2021 | 122.0 | 0.000 | 122.122807 | 0.5691527 | 121.0 | 124.0 | 0.47 | 57 |
| JBGAS | BGLUC | RAD1 | R0173 EXP MAR 2021 | 1.5 | 0.000 | 1.507017 | 0.0257713 | 1.5 | 1.6 | 1.71 | 57 |
| JBGAS | BGLUC | RAD2 | R0175 EXP MAR 2021 | 5.6 | 0.075 | 5.585897 | 0.0639081 | 5.4 | 5.7 | 1.14 | 78 |
| JBGAS | BGLUC | RAD3 | R0141 EXP SEP 2017 | 13.7 | 0.100 | 13.763636 | 0.1310409 | 13.4 | 14.1 | 0.95 | 55 |

And there we have it:

Now I can write the output file

write\_csv(summary.table.no.outliers, "QC\_summary.csv")

With dplyr, if you direct your energies to the right place, you reap much. Similarly:

*“But seek ye first the kingdom of God, and his righteousness; and all these things shall be added unto you.”*

*Matthew 6:33*